Ad Hoc & Sensor Wireless Networks Vol. 9, pp. 85–107
Reprints available directly from the publisher
Photocopying permitted by license only

Query Dissemination in Sensor Networks – Predicting Reachability and Energy Consumption

MARKUS BESTEHORN[2], ZINAIDA BENENSON[1,*], ERIK BUCHMANN[2], MAREK JAWUREK[3], KLEMENS BÖHM[2] AND FELIX C. FREILING[1]

[1]*University of Mannheim, Germany*
[2]*Universität Karlsruhe (TH), Germany*
[3]*Fraunhofer IESE, Germany*

Received: October 22, 2008. Accepted: December 3, 2008.

Specifying information needs declaratively has turned out to be useful in the context of databases systems, and reseachers have applied this approach to sensor networks. The first step to retrieve data from a sensor network in this way is the dissemination of the query. Energy-efficient query dissemination plays an important role for the lifetime of sensor networks. The topic of this paper is the optimization of probabilistic query dissemination: Based on easy-to-acquire topology information, the optimizer can predict the number of nodes reached and the energy consumption. Furthermore, we propose a topology-discovery protocol that collects the structural information required by the optimizer. Our analysis shows that the energy savings exceed the energy spent to obtain the required information after a small number of queries disseminated in realistic settings. Our evaluation explores the tradeoff between reachability and energy consumption using both simulations and a deployment of Sun SPOT sensor nodes.

Keywords: Wireless sensor networks, probabilistic broadcasting, query dissemination, broadcast optimization, energy efficiency.

1 INTRODUCTION

Wireless sensor networks have been established in many important application areas from ambient intelligence over scientific research to industrial

*Zinaida Benenson was supported by Landesstiftung Baden Württemberg as part of Project "ZeuS" and by the Schlieben-Lange scholarship of the European Social Fund and the Bundesland Baden-Würtemberg.

uses. Such sensor networks usually consist of numerous battery-powered, stationary nodes [1, 2] equipped with sensing devices, low-power wireless communication and limited computational resources. In order to fulfill complex measurement tasks, the sensor nodes use self-organization techniques to form ad-hoc networks.

State-of-the-art query processors [3, 4] allow the users to access sensor information by issuing declarative queries. Processing of such queries consists of 4 phases, in which the nodes (1) forward the query from a central base station to all nodes, (2) capture sensor values, (3) do in-network query processing and (4) return the results to the base station. Since nodes are battery powered, energy is a valuable resource. Thus, all of these steps must be accomplished using as little energy as possible.

As sending, receiving and idle listening are the most energy-intensive operations, reducing communication is the most important optimization goal in sensor networks. While a lot of research, e.g., [3–5], has focused on Phases (2) to (4), the query-dissemination phase has received much less attention. However, since a query must reach all nodes in the network, while the query result might originate from a few nodes only, the first phase causes much communication. To avoid unnecessary energy consumption for query dissemination, we focus on this first phase of query processing.

One way to save communication effort is to reduce the number of query broadcasts in the dissemination phase. However, this may have a negative effect on quality-of-service parameters of the query. If energy is saved by querying only, say, 50% of the nodes, the accuracy of the query result degrades. How much it degrades depends on many factors, e.g., the query, the distribution of the nodes reached, and correlations between values measured at different nodes. The relation between these factors, the quality of the result and the percentage of nodes reached is not very well understood. Quantifying this tradeoff between the dissemination strategy and the service quality is the topic of this paper.

Related Work. While numerous sophisticated in-network query processing techniques have been developed [3, 4, 6, 7], they mostly focus on operator processing, optimization and aggregation techniques. The dissemination of the query from the base station into the network has either been disregarded or is done via simple flooding [5, 8]. It is well known that flooding wastes energy. For example, analyses [9] have shown that a rebroadcast increases the area where the message is received by 61% at most, dropping to $\approx$20% for average networks, i.e., most rebroadcasts will not let additional nodes receive the query if simple flooding is used. Furthermore, most nodes receive the query more than once. This should be avoided, since receiving messages consumes energy as well.

To avoid the disadvantages of simple flooding, several mechanisms for broadcasting in wireless networks have been proposed. An overview on such

broadcasting mechanisms is presented in [10]. Generally, these approaches try to control which nodes rebroadcast a message in order to keep the number of nodes that receive the query more than once as small as possible.

In *neighbor-knowledge*-broadcasting schemes [11, 12], nodes use detailed local topology information to determine which nodes must rebroadcast a message: For example, [11] requires every node to store topology information about all nodes reachable via 2 hops. This information is used to ensure that a node rebroadcasts only if this is necessary to reach all nodes. However, this comes at cost of acquiring and maintaining topology information, which is significant. Furthermore, identifying the nodes which have to broadcast is difficult, even if full topology information is available: Finding a minimal set of rebroadcasting nodes is equivalent to the Dominating Set Problem, which is NP-complete [13].

Another promising category of approaches for broadcasting in wireless networks are *probabilistic* or *epidemic* algorithms. Compared to schemes using neighbor-knowledge, these methods do not need to acquire, store and update knowledge about nodes in their vicinity. For example, with *counter-based flooding* [9, 14], if a node hears k or more of its neighbors rebroadcast the message, it suppresses its own transmission. Similarly, probabilistic approaches assign a probability p to each node which determines the probability of a rebroadcast. To determine the value of the afore mentioned parameters, these approaches also require some, but less detailed knowledge about the network. For example, if the rebroadcast probability p is too high, the disadvantages of simple flooding arise, and if p is too low, only a fraction of nodes receive the broadcast message.

Contributions. In this paper, we study query-dissemination techniques that are a combination of acquiring topological information, reachability estimation and probabilistic flooding. We develop a model which predicts the number of nodes reached and the energy consumed based on relatively coarse topology information, compared to topology-based approaches, and a given rebroadcast probability. Using extensive simulations and experiments we explore the tradeoff between energy, reachability and structural information required. We show that little structural information on the network allows to predict the number of nodes reached according to a certain broadcast probability p. Our model allows estimating the number of transmissions in advance. More specifically, we make the following contributions:

1. We introduce a model to estimate the reachability and the number of transmissions depending on to the rebroadcast probability p. Our model relies on connectivity information and requires a histogram containing the number of nodes reached with each rebroadcast, starting at the base station.

2. We discuss several options to acquire the topology information required for our approach, and we describe a lightweight topology-discovery protocol to obtain this information. Our analysis shows that gathering structural information and computing an optimal p saves energy after a small number of probabilistic floodings in realistic settings.

3. We have conducted simulations with up to 425 nodes to verify the results of our model for large numbers of nodes. Furthermore, we have tested our findings using a real testbed consisting of 17 Sun Spot sensor nodes.

Outline. In Section 2 we present a model which estimates the number of nodes reached and energy spent by probabilistic flooding for a particular rebroadcast probability p. The model depends on topological information. In Section 3 we show how to gather the information required efficiently. In Section 4 we present simulation and experimental results, and we conclude in Section 5.

2 REACHABILITY AND ENERGY-CONSUMPTION PREDICTION FOR QUERY DISSEMINATION

In this work we focus on probabilistic flooding where each node rebroadcasts queries with a fixed probability p. Parameter p allows to fine-tune the trade-off between energy spent for query dissemination and the number of nodes reached. Moreover, the analysis in [15] has shown that in most (densely connected) sensor networks there exists a $p_0 < 1$ such that all nodes are reached by the base station. Thus, if the rebroadcast probability p is larger than p_0, more queries are forwarded than necessary, and the query dissemination can save energy by using p_0. On the other hand, if $p < p_0$, the query reaches only a fraction of nodes. This can be useful to trade energy for result quality. Our goal is to develop a model to predict for every p the number of nodes R reached and the energy E consumed by the query-dissemination process. Knowing the dependencies between p, R and E allows the base station to estimate how many nodes can be reached using a fixed amount of energy, or at which p improving the reachability means spending a huge amount of energy to reach only a few nodes more.

Energy-usage prediction depends on reachability prediction, which in turn depends on the network topology. The more the base station knows about network topology, the more precise the predictions can be. However, gathering topology information consumes energy. We are interested in making predictions using topological information which can be obtained without exhausting potential energy savings due to gathering fine-grained topology information. In the following we will predict the reachability $R(p)$ and energy consumption $E(p)$ according to given topological information and a rebroadcast probability p. More specifically, $R(p)$ estimates the number of nodes reached, and

$E(p)$ estimates the energy consumption based on the number of messages sent and received.

2.1 Assumptions and notations

Our estimation of the reachability relies on two assumptions:

A1 The network is in a stable state while flooding the query, i.e., the topology does not change between obtaining topology information and flooding.

A2 A node is either reached by a node that is one hop closer to the base station, or by a node that has the same hop distance to the base station.

The rationale behind Assumption A1 is that, if the topology differs significantly from the topology information available for reachability prediction, the accuracy of the prediction degrades. In reality, networks change over time due to nodes failing or external factors, e.g., weather, improving or degrading radio communication reception. Hence, the topology information may be different from the situation in reality. Thus, Assumption A1 relies on the presence of a mechanism that keeps the topology information sufficiently accurate. In Section 3 we propose such a mechanism and therefore do not factor differences between topology information and reality into our prediction model.

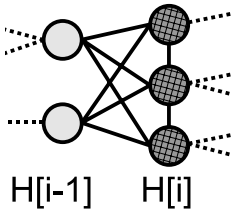
During query dissemination, the messages containing the query disperse through a topology in several steps, beginning at the base station. The nodes which receive the query from the base station can rebroadcast it, so that the query may reach nodes two hops away from the base station. This procedure recurs until all nodes that are connected to the network have received the query. To avoid unnecessary broadcasts, each node stores an identifier for all messages received and does not rebroadcast messages with a known identifier.

If a Node A receives a broadcast message from a Node B, we say that A is reached by B. We will denote the set of all nodes reached with h hops as *hop set* $H[h]$. Hop sets are disjoint, i.e., each node is member of exactly one hop set. Note that our definition of hop set does not depend on assumptions that do not hold in reality [16], specifically geographical neighborhood, symmetric links or regular communication ranges between the nodes.

2.2 Topological information

Our analytical model depends on the following topological information. Section 3 will introduce a protocol that collects this information efficiently:

- *histogram*$[h]$: stores the number of nodes reached at each hop from the base station, i.e., $\forall i \in \{1 \cdots n\} : \mathit{histogram}[i] = |H[i]|$.
- *connectivity*$[h]$ stores the average number of connections from one node in hop set $H[h]$ to a node from $H[h-1]$.



(a) Example hop sets

…	$i-1$	i	…
…	2	3	…

(b) histogram

…	$i-1$	i	…
…	1.5	2	…

(c) connectivity

…	$i-1$	i	…
…	0	$4/3$	…

(d) interconnectivity

FIGURE 1
Example hops sets and their histogram representation.

– *interconnectivity* $[h]$ stores the average number of connections between the nodes from the same hop set, i.e., the connections a node in $H[h]$ has to other nodes in $H[h]$.

In the example in Figure 1(a) the hop set $H[i]$ consists of 3 nodes, the previous hop set $H[i-1]$ consists of 2 nodes. Edges represent bi-directional links between nodes. Figure 1 shows the histogram, connectivity and interconnectivity for the example in Figure 1(a): Recall that *histogram* $[h]$ is the number of nodes in hop set $H[h]$, i.e., hop set $H[i]$ contains 3 nodes in figure 1, thus *histogram* $[i] = 3$. The value for interconnectivity equals the average number of connections within a hop set, e.g., for the example in Figure 1 *interconnectivity* $[i]$ is computed as follows: The node on top has a link to the node in the middle, the node in the middle a connection to the node on top and to the bottom one, and the node at the bottom a connection the node in the middle. Thus there are 4 connections within $H[i]$ and 3 nodes. This results in *interconnectivity* $[i] = \frac{4}{3}$. Connectivity is computed similarly.

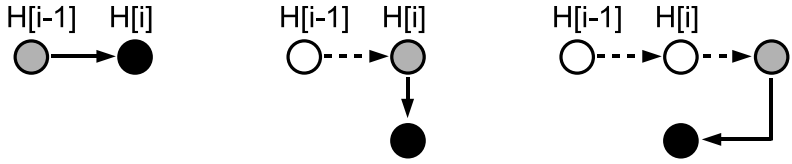
2.3 Reachability prediction

Using the abstraction of the hop set, a node can receive a message broadcast by another node in different ways, as illustrated in Figure 2:

Direct Reachability: A node of hop set $H[i]$ is *reached directly* if the sender of the broadcast is a member of hop set $H[i-1]$ (Figure 2(a)).

Indirect Reachability: A node in hop set $H[i]$ is *reached indirectly* if the sender of the broadcast is a member of the hop set $H[i]$ as well (Figure 2(b)).

Backward Reachability: A node in $H[i]$ is *reached backwards* if the sender of the broadcast is a member of a hop set $H[j]$ with $j > i$ (Figure 2(c)).



(a) Direct Reachability (b) Indirect Reachability (c) Backwards Reachability

FIGURE 2
Possibilities for reaching a node via broadcast.

Taking backward reachability into account would require more detailed topology information which would have to be gathered by spending more energy. Hence, we will not consider backward transmissions. We will show in Section 4, that our predictions are sufficiently accurate to determine a probability where (almost) all nodes are reached, but only a fraction of nodes forwards the queries.

For the prediction of $R(p)$, the number of nodes reached with rebroadcast probability p, let $R_{direct}(h, p)$ be the number of nodes in hop set $H[h]$ which have been reached directly, and let $R_{indirect}(h, p)$ denote the number of nodes which are reached indirectly. Since backward reachability is left aside, the number of nodes $R(h, p)$ reached at the h-th hop can be computed as follows:

$$R(h, p) = min\left(R_{direct}(h, p) + R_{indirect}(h, p), histogram[h]\right) \quad (1)$$

In case of simple flooding, i.e., $p = 1$, the predicted number of nodes $R(h, 1)$ is the number of nodes in the hop set $H[h]$. If p is set to a low value, certain nodes do not rebroadcast the query, and therefore $R_{direct}(h, p) + R_{indirect}(h, p)$ can be less than $histogram[h]$. Since a node can receive the same query more than once from surrounding nodes, $R_{direct}(h, p) + R_{indirect}(h, p)$ can be larger than the number of nodes in the hop set $H[h]$. The minimum function ensures that at most the actual number of nodes in the hop set is returned. The total reachability for a given p is the sum over all hops:

$$R(p) = \sum_{h} R(h, p) \quad (2)$$

In the following we will show how the functions $R_{direct}(h, p)$ and $R_{indirect}(h, p)$ can be computed to predict $R(h, p)$.

Predicting direct reachability. Since the base station always forwards, all nodes in the first hop set $H[1]$ receive the query, i.e., $R(1, p) = histogram[1]$. For all $h > 1$, $R(h, p)$ can be computed recursively using the following idea: A node in the previous hop set $H[h]$ can only rebroadcast if it has been reached. Therefore, the number of nodes that can rebroadcast the message from $H[h-1]$ to $H[h]$ is equal to the number of nodes reached in the previous

hop set $H\,[h-1]$, i.e., $R\,(h-1,\,p)$. Of these $R\,(h-1,\,p)$ nodes, only $k = R\,(h-1,\,p)\cdot p$ rebroadcast.

Let P (event) denote the probability of a certain event. Now we need the probability of the event "A node from hop set $H\,[h]$ receives its message from a node from hop set $H\,[h-1]$". The probability of this event is:

$$P\text{ (reached directly)} = 1 - P\text{ (not reached directly)} \tag{3}$$

We compute the counter-event "not reached directly" by considering the nodes in $H\,[h-1]$ which have not received the query previously. Let $m = \mathit{histogram}\,[h-1]$, and recall that $k = R\,(h-1,\,p)\cdot p$, then $m-k$ nodes in $H\,[h-1]$ do not rebroadcast. Thus the hop set $H\,[h-1]$ is partitioned into a set of k broadcasters and $m-k$ non-broadcasters. The fundamental idea of Equation 4, which computes the probability of the counter-event, is as follows: The counter-event corresponds to randomly choosing *connectivity* $[h]$ nodes out of hop set $H\,[h-1]$ and choosing non-broadcasters only. When randomly choosing the first node, the probability to choose one of the $m-k$ non-broadcasters is $\frac{m-k}{m}$. For every node chosen, the total number of nodes remaining is reduced by 1. Therefore, assuming that the first node chosen was a non-broadcaster, the probability the next randomly chosen node is also a non-broadcaster is $\frac{m-k-1}{m-1}$. Thus, the probability that the first *connectivity* $[h]$ randomly chosen nodes are non-broadcasters is computed by Equation 4:

$$P\text{ (not reached directly)} = \prod_{l=0}^{\lceil \mathit{connectivity}[h]\rceil - 1} \frac{m-k-l}{m-l} \tag{4}$$

We now calculate the number of nodes from hop set $H\,[h]$ receiving the query directly by multiplying the probability P (reached directly) with the number of nodes in the hop set:

$$R_{\mathit{direct}}\,(h,\,p) = P\text{ (reached directly)}\cdot \mathit{histogram}\,[h] \quad \text{for h>1} \tag{5}$$

Due to the fact that P (reached directly) is computed using $k = R\,(h-1,\,p)\cdot p$, this results in the following recursive function:

$$R_{\mathit{direct}}\,(h,\,p) = \begin{cases} \mathit{histogram}\,[1] & \text{if } h = 1 \\ P\text{ (reached directly)}\cdot \mathit{histogram}\,[h] & \text{if } h > 1 \end{cases} \tag{6}$$

The remaining nodes in hop set $H\,[h]$ can still be reached indirectly, i.e., by a subsequent broadcast by nodes from the same hop set.

Predicting indirect reachability. To calculate the number of nodes reached indirectly, we assume that the nodes which have received the message are equally distributed over the hop set. Thus, if k from n nodes are directly reached, each node in the hop set has obtained the message with probability

$\frac{k}{n}$. Our experimental evaluation will show that this simplification is legitimate, i.e., it is not necessary to collect topological information in more detail. To estimate the number of nodes reached indirectly, we multiply the probability P (reached directly) $\cdot\, p$ that a node reached directly rebroadcasts with the interconnectivity and number of nodes reached in the hop set:

$$R_{indirect}(h, p) = P(\text{reached directly}) \cdot p \cdot interconnectivity[h] \cdot histogram[h] \tag{7}$$

2.4 Prediction of energy consumption

Having estimated the number of nodes reached, we will estimate the energy required by probabilistic flooding. Therefore, we distinguish between messages sent and received. The number of messages sent in hop set $H[h]$ is as follows:

$$msgs_{sent}(h, p) = R(h, p) \cdot p \tag{8}$$

A node in hop set $H[h]$ might receive messages from nodes in many other hop sets. As we have explained in Subsection 2.3, modeling backward reachability would require topological information with a very high level of detail. We estimate the number of received messages $msgs_{received}(h, p)$ by considering nodes in the previous ($H[h-1]$), current ($H[h]$) and next ($H[h+1]$) hop set, because the majority of broadcasts are received from these nodes. Equations 9 and 10 estimate how many messages sent by nodes in $H[h]$ are received from nodes in the previous/current hop set:

$$rec_{previous}(h, p) = msgs_{sent}(h, p) \cdot connectivity[h] \tag{9}$$

$$rec_{current}(h, p) = msgs_{sent}(h, p) \cdot interconnectivity[h] \tag{10}$$

Since topology information on backward reachability is not available, we estimate the number of connections from $H[h+1]$ to $H[h]$ based on the average connectivity of nodes in $H[h]$ to nodes $H[h+1]$. Thus, we assume the number of forward connections the next hop set is equivalent to the number of backward connections. We multiply this number with the number of sent messages to estimate the number of received messages in the next hop set $rec_{next}(h, p)$:

$$rec_{next}(h, p) = msgs_{sent}(h, p) \cdot \frac{connectivity[h+1] \cdot histogram[h+1]}{histogram[h]} \tag{11}$$

Finally, the total number of received messages can be estimated as

$$msgs_{received}(h, p) = rec_{previous}(h, p) + rec_{current}(h, p) + rec_{next}(h, p) \tag{12}$$

The total energy cost of the probabilistic flooding is calculated by multiplying the messages sent and received with the vector of energy costs for sending and

Name	Description	Data Type
sender	Sender of the TDReq	integer
hop	Hop Number	integer

FIGURE 3
Contents of a Topology-Discovery Request Message.

receiving, and adding them up for every hop set:

$$E(p) = \sum_{h} (msgs_{sent}, msgs_{received})_{(h,p)} \cdot \begin{pmatrix} energyPerSend \\ energyPerReceive \end{pmatrix} \qquad (13)$$

3 ACQUIRING TOPOLOGY INFORMATION

Our prediction model requires (1) a histogram containing the size of each hop set, (2) a list of values which specify the connectivity between hop sets and (3) a list of values which specify the connectivity within every hop set. There are several ways to acquire such information, e.g., by using gossip protocols or by extracting topology information from meta-data of the routing protocol [17–19]. However, these approaches strongly depend on the underlying communication protocols, hardware and system architecture. To avoid such dependencies, we decided to develop a lightweight topology discovery protocol based on the principle of the *echo algorithm* [20] and used it during our evaluation.

3.1 Echo-based topology-discovery

The *echo algorithm* [20] consists of an *expansion* wave, where messages are flooded from the base station to distant nodes, and a *contraction* wave that flows back to the base station. We use this concept to transport a request for topology information in the expansion wave, and we aggregate and return this information in the contraction wave. In particular, the base station initiates the topology discovery by broadcasting a *Topology-Discovery Request Message* (TDReq), thus starting the expansion wave. Figure 3 describes a TDReq message.

The small network in Figure 4 will illustrate the topology-discovery process. Edges between nodes in Figure 4 represent bi-directional links, and the square in the middle represents the base station. In this example, the topology discovery is started by the base station broadcasting a TDReq to Nodes 1, 2, 3 and 4.

Expansion wave. When a node ℵ receives a TDReq for the first time, the receiver must accomplish 4 steps, as illustrated in Algorithm 1:

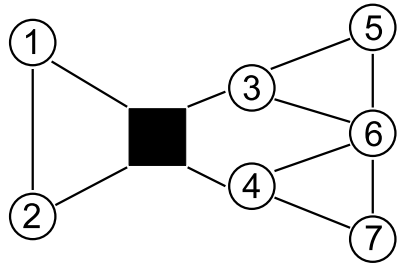


FIGURE 4
Example network to illustrate the topology-discovery process.

1. Create three empty node lists *uncles*, *siblings* and *children* (Line 3) and mark the *sender* of the `TDReq` as the parent node of ℵ (Line 5). ℵ also extracts the hop number h from the `TDReq` and stores it (Line 4).
2. Broadcast own request message with an incremented hop number. The *sender* value must be the ID of ℵ (Lines 6–8).
3. Start a timeout (Line 9) to ensure that ℵ does not wait forever for potential children. The length of the timeout should be sufficiently long to allow the children to receive, process and rebroadcast the Topology-Discovery Request.
4. Wait until the timeout expires or messages belonging to the contraction wave have been received from all children, then start the contraction wave.

In the example in Figure 4, Nodes 1, 2, 3 and 4 receive the `TDReq` from the base station, thus mark the base station as their parent node, rebroadcast the `TDReq` with an incremented hop counter and start a timeout to wait for further `TDReq` (cf. Lines 5–9 of Algorithm 1).

After receiving the first `TDReq` and while waiting for the timeout of Line 9, a node can receive further `TDReq` messages, since every node broadcasts its own `TDReq` message (see Step 2 above). As illustrated in Figure 2, any broadcast can reach a node in 3 ways: direct, indirect and backwards. Depending on the originator of the `TDReq` message, a node that receives a `TDReq` message must modify one these lists:

– **Uncles**: This case corresponds to the direct reachability case where a node with a distance of h hops to the base station receives a `TDReq` from a node with distance $h - 1$. The first `TDReq` received is from the parent node in the previous hop set $H[h - 1]$. Every further `TDReq` corresponds to an additional connection from the h-th hop set to the

Algorithm 1: Handling of incoming `TDReq` messages.

```
1  upon receive:TopologyDiscoveryRequest req
2      if req is the first TDReq received then
3          create empty lists uncles, siblings and children;
4          myhop = req.hop;
5          myparent = req.sender;
6          req.hop++;
7          req.sender = this;
8          rebroadcast(req);
9          startTimeout(NoChildren);
10     else
11         if req.hop == myhop+2 and req.parent = this then
12             children += req.sender
13         if req.hop == myhop+1 then
14             siblings += req.sender
15         if req.hop == myhop then
16             uncles += req.sender
17
```

previous hop set $h-1$. Line 16 adds every sender of all `TDReq` after the first `TDReq` to *uncles*.

- **Siblings**: Corresponding to the case of indirect reachability, this occurs when a node with a distance of h hops to the base station receives a `TDReq` message from a node with the same distance to the base station. The sender of the node thus is a sibling of the receiver and therefore must be added to the *siblings* list (Line 14).
- **Children**: This corresponds to backwards reachability, where a node receives a `TDReq` from a node whose distance to the base station is larger. If id_{parent} contained in such a `TDReq` is the identifier of the current node, the sender is a child and thus is inserted into the *children* list. If id_{parent} does not equal the identifier of the current node, the message can be ignored to avoid that children are counted multiple times by nodes in $H[h]$ (Line 12).

In the example in Figure 4, Node 1 receives a `TDReq` message from Node 2. Since Node 2 has incremented the hop counter before rebroadcasting, Node 1 adds Node 2 to its *siblings* list and continues to wait until the timeout in Line 9 expires. Similarly, Node 2 will add Node 1 as a sibling and start the contraction wave once the timeout expired, sine both Node 1 and 2 do not have any children. Assuming that Node 3 rebroadcasts the `TDReq` message prior to Node 4, Node 3 is parent of Nodes 5 and 6. When Node 3 receives `TDReq`

messages from Nodes 5 and 6, both are added to the *children* list. Once Node 4 rebroadcasts its `TDReq` message, Node 6 will receive this message and add Node 4 to the *uncles* list.

Algorithm 2 shows the possible transitions from the expansion to the contraction wave.

Leaf nodes like Nodes 1, 2, 5, 6 and 7 in Figure 4 will wait until their timeout expires and initiate the contraction phase, since their list of children is empty (cf. Line 5 in Algorithm 2). The list of children is only used to determine when the topology information of all children has been collected, i.e., Node 3 will wait for Nodes 5 and 6 to send their aggregated topology information until it compiles its topology information and sends it to the base station/parent node (Line 10 of Algorithm 2). The initiation of the contraction wave and the aggregation of topology information is described next.

Contraction wave. The contraction wave starts on leaf nodes where the timeout (cf. Step 3 above) expires without any incoming response messages. After the timeout has expired, the leaf node will execute Algorithm 3 in order to initialize the data structures required.

After the initialization, the leaf node creates a *Topology-Discovery Response Message* (`TDResp`) containing these lists. This response message

Algorithm 2: Transitions from expansion to contraction phase.

```
1  upon timeout NoChildren
2      if  children != {} then
3          startTimeout(WaitForChildren)
4      if children == {} then
5          msg=compileInformationLeaf();
6          send msg to parent;
7
8  upon timeout WaitForChildren or AllChildrenResponded
9      msg=compileInformationFromChildren();
10     send msg to parent;
11
```

Algorithm 3: Algorithm for histogram initialization at leaf nodes.

```
   /* Called if node is a leaf node                    */
1  method compileInformationLeaf
2      histogram[0]=1;
3      connectivity=(count(uncles)+1,1);
4      interconnectivity=(count(siblings),1);
5      return histogram,connectivity,interconnectivity;
6
```

is sent from every leaf node, e.g., Nodes 1, 2, 5, 6 and 7 in Figure 4, to its corresponding parent node, i.e., the node from which the first `TDReq` has been received.

In case the node has children, i.e., *children* is not empty, the node will wait for all children to return their response message (Line 8 in Algorithm 2). To avoid endless waiting because a child fails to return its response message, we limit the waiting time for children with a sufficiently large timeout. When either this timeout expires, or if all children have responded, the lists of all children must be aggregated as illustrated in Algorithm 4:

Algorithm 4: Methods histogram creation and aggregation.

```
   /* Called if node is a non-leaf node                    */
 1 method compileInformationFromChildren
 2     foreach child do
 3         combine child.histogram into Histogram;
 4         combine child.connectivity into Connectivity;
 5         combine child.interconnectivity into InterConnectivity;
 6     shift Histogram by 1 to the right;
 7     shift Connectivity by 1 to the right;
 8     shift Interconnectivity by 1 to the right;
 9     Histogram[0]=1;
10     Connectivity[0]=(count(uncles) + 1,1);
11     InterConnectivitgy[0]=(count(siblings),1);
12     return Histogram,Connectivity,InterConnectivity;
13
```

The lists of the child nodes are combined by adding the entries, i.e., Position *i* of the resulting lists contains the sum of the list entries of the children (Lines 2–5). In Lines 6 to 8 the entries are shifted by 1 entry to the right to make space for the list entries of the current node. As a last step, the topology information of the current node is aggregated by counting uncles and siblings (Lines 9–11) and written into the first position of the list. The results are sent to the parent node of the current node, as described in Line 10 of Algorithm 2.

Computing the topology information at the base station. After the base station receives all Topology-Discovery Responses from its children, it aggregates information in the same way as described in Algorithm 4, but transforms the connectivity and interconnectivity lists into floating point numbers. Every tuple is transformed by dividing the first entry by the second, resulting in the average number of connections per node in the specific hop set. Histogram, connectivity and interconnectivity are stored locally for the reachability-prediction algorithm.

Energy cost and message size for Topology-Discovery Protocol. As every node broadcasts one Topology-Discovery Request and sends one Topology-Discovery Response, the energy costs per node are:

$$E_{Node} = E_{send}(b_1) + E_{send}(b_2) + AverageNodeDegree * (E_{rcv}(b_1) + E_{rcv}(b_2)) \quad (14)$$

The value b_1 stands for the number of bytes in the Topology-Discovery Request of the node, b_2 is the number of bytes in the Topology-Discovery Response. In Section 4.1 we will calculate energy consumption of the Topology-Discovery Protocol, and we will show that it pays off after a few query disseminations to collect this information in order to estimate a "good" p.

4 EVALUATION

In this section we evaluate the prediction model with different node setups using simulations and a deployment of 17 Sun SPOT sensor nodes [2]. We compare the predictions to the query dissemination in simulated networks of up to 425 nodes and in a real sensor network. We show the following:

1. For all networks simulated and the real sensor network, the accuracy of the reachability prediction based on the topology information is high.
2. The energy saved clearly outweighs the inaccuracy related to the probabilistic flooding.

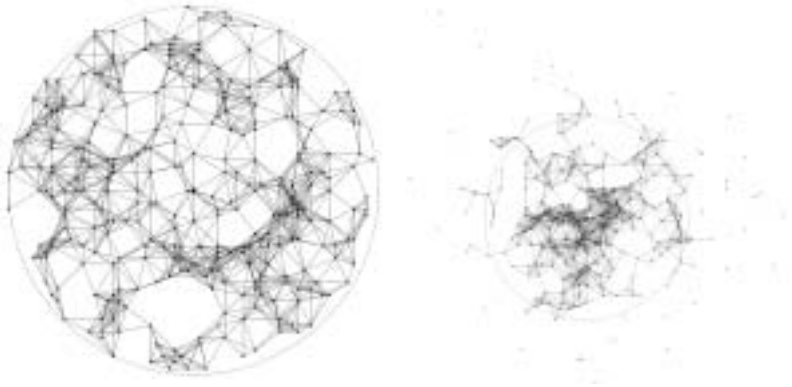
Our model produces stochastic results for the average case, i.e., it works well for sufficiently dense networks or for large numbers of trials. Thus, we expect a small deviation between the predicted values and experimental results.

4.1 Simulations

For the simulation we used our *Karlsruhe Sensor Networking Simulator* [21], which is interface-compatible to Sun SPOT sensor nodes. This enables us to deploy the prediction model and the topology-discovery protocol in both the simulated environment and the real deployment.

Simulation setup. To evaluate our approach with a wide range of parameters, we generated networks of varying node densities and two different topologies.

1. **Uniform**: This topology is an example for a sensor deployment that has been carefully planned to provide a defined coverage of a region. The nodes are distributed uniformly in a circular area with a radius of 30 units around the base station, as illustrated in Figure 5(a). We used a fixed radius and varied the average number of neighbors (the *average*



(a) Example: Uniform Topology (Node Degree 12)

(b) Example: Gaussian Topology (325 Nodes)

FIGURE 5
Exemplary node distributions with 325 nodes.

node degree) for every node from 4 to 16 to create topologies ranging from sparse to dense.

2. **Gaussian**: This topology corresponds to a "smart-dust" scenario where the nodes are arbitrarily poured into the area, e.g., from an airplane flying over a forest fire. The placement of nodes follows a Gaussian distribution. In particular, we use Gaussian sampling with the center of the environment as mean and a standard deviation of 18 units to place the nodes. Again, the area covered has a size of 30 units. For our simulations, we vary the number of nodes from 125 to 425. As shown in in Figure 5(b), most nodes are located close to the center, and the further away from the base station, the lower the node density. Because some of the nodes close to the edge of the area are disconnected from the network, even a rebroadcast probability of $p = 1.0$ will not deliver the query to all nodes.

We generated 40 instances for each parameter setup to exclude stochastic errors. To enable a comparison between both topology types, Table 1 shows

Average Node Degree	Used Sensors
4	125
8	225
12	325
16	425

TABLE 1
Average node degree in Uniform scenario and number of nodes

which node degree in uniform node distributions equals which number of nodes.

Energy consumption parameters. In order to determine the energy consumption for (1) a simulated query dissemination and (2) the energy-consumption prediction, the costs of sending and receiving a message of a certain size must be calculated. We obtained the energy consumption from an analysis [22] of MICAz [1] nodes. [22] measured the energy consumption for standard TinyOS [23] messages with a payload b of up to 28 bytes sent/received:

$$\begin{aligned} \mathit{EnergyForSending}(b) &= 0.185191\ \text{mAs} + (b - 28\mathit{byte}) \\ &\quad * 2.48461\ \text{mAs} * 10^{-5} \end{aligned} \tag{15}$$

$$\begin{aligned} \mathit{EnergyForReceiving}(b) &= 0.042\ \text{mAs} + (b - 28\mathit{byte}) \\ &\quad * 2.47915\ \text{mAs} * 10^{-5} \end{aligned} \tag{16}$$

Experiment execution. We evaluated each topology in four steps:

Step 1 Fetch the required topology information using the topology-discovery protocol of Section 3.

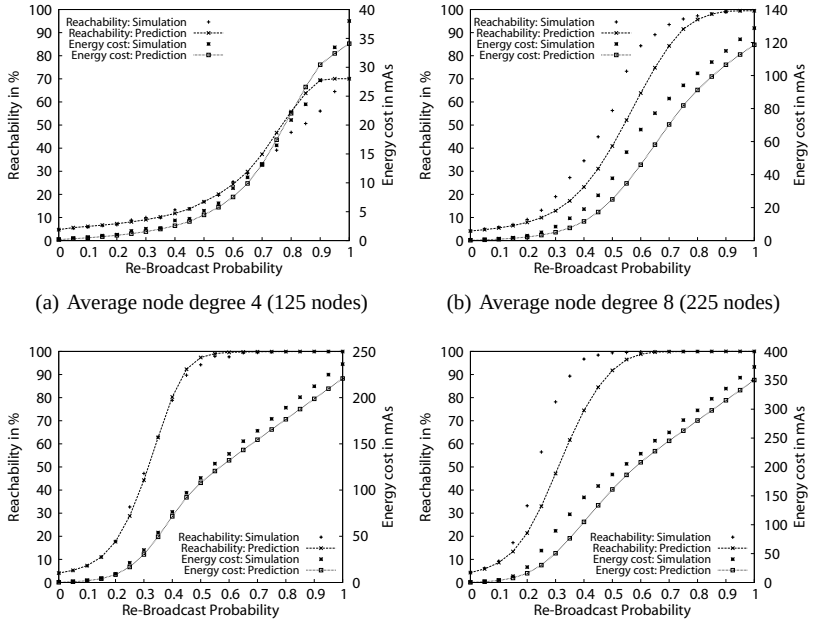
Step 2 Predict the total reachability $R(p)$ for rebroadcast probabilities $p = \{0, 0.05, \ldots, 1\}$.

Step 3 For each rebroadcast probability, simulate 120 query disseminations and count the number of messages sent and received.

Step 4 Compute the energy consumption using Equations (15) and (16).

Simulation results. Figures 6 and 7 show the results of our reachability and energy-consumption prediction for uniform/Gaussian node distributions. The energy savings are large. Consider Figure 6(d). Simple flooding ($p = 1$) consumes about 370 mAs. According to our model, all nodes are reached with $p_0 = 0.6$. Thus, we save 37% energy as compared to rebroadcasting at every node.

Our prediction is good in sparse networks, i.e., for 125 nodes or an average node degree of 4. For dense networks, our prediction tends to underestimate the reachability and energy consumption. The reason for this is that our model does not consider backwards reachability: In dense networks, many nodes receive a broadcast from nodes that are more hops away from the base station than the receiver. Due to the fact that our model neglects these transmissions, we estimate a number of nodes reached that is smaller than observed in the simulations. However, this does not limit the applicability of our model. When calculating the rebroadcast probability p from an underestimation, p is slightly larger than necessary, i.e., it includes some margin of safety, and the predicted



(a) Average node degree 4 (125 nodes)

(b) Average node degree 8 (225 nodes)

(c) Average node degree 12 (325 nodes)

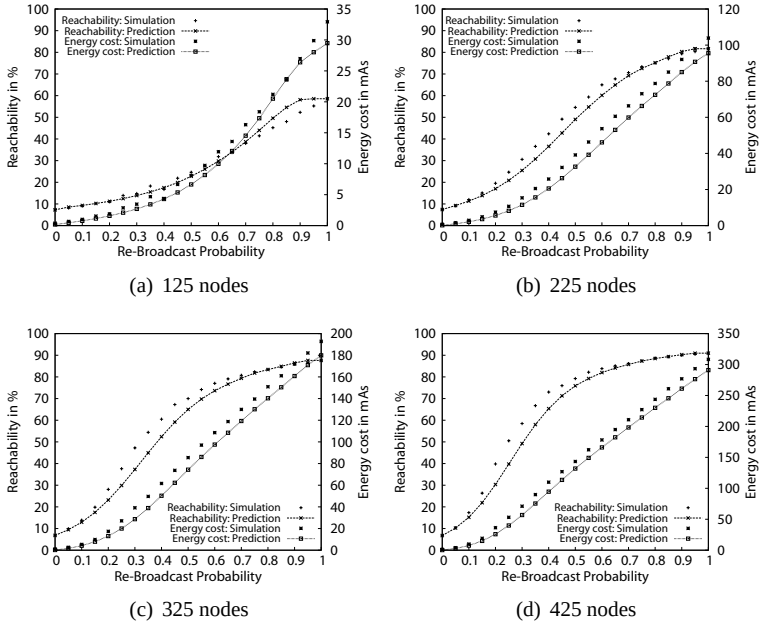
(d) Average node degree 16 (425 nodes)

FIGURE 6
Comparison of simulated reachability/energy cost in uniform scenarios.

number of nodes should be reached. As Figure 7 shows, the predictions for the Gaussian scenarios are more accurate than the ones for the uniform scenarios. Recall that in Gaussian scenarios, some nodes are placed so far from the base station that the network becomes disconnected.

Topology-Discovery and Reachability-Prediction payoff. Now we will show that the topology-discovery pays off after a a few query disseminations. Assume a uniform scenario with 425 nodes, average node degree 16 and a reachability of about 99%. In this scenario, our prediction of p_0 saves 150 mAs (see Figure 6(d)): Using a rebroadcast probability of $p = 0.6$, 220 mAs are consumed to reach all nodes. In comparison, simple flooding ($p = 1$) consumes 370 mAs. However, the topology discovery requires energy as well. According to Formulae (14)–(16), the Topology-Discovery Protocol consumes 722 mAs:

$$\begin{aligned} E_{TopDisc} &= 0.1871 \frac{\text{mAs}}{Broadcast} \cdot 425\ Node \cdot 2 \frac{Broadcasts}{Node} \\ &\quad + 0.0414 \frac{\text{mAs}}{Receive} \cdot 425\ Node \cdot 2 \frac{Broadcasts}{Node} \cdot 16 \frac{Receive}{Broadcast} \\ &= 722.0745\ \text{mAs} \end{aligned} \tag{17}$$



(a) 125 nodes (b) 225 nodes

(c) 325 nodes (d) 425 nodes

FIGURE 7
Comparison of simulated reachability/energy cost in Gauss scenarios.

In this particular uniform scenario, topology discovery pays off after 5 query disseminations. A calculation for the Gaussian topology yields similar results. Generally the topology discovery will pay off for any kind of densely connected sensor network after a few query disseminations, compared to simple flooding. As the analysis in [15] has shown, there exists a $p_0 < 1$ such that probabilistic flooding reaches all nodes for most sensor networks. Our prediction model allows to precompute p_0 and then disseminate the query using a rebroadcast probability close or equal to p_0, thus saving energy.

4.2 Sun SPOT deployment

We have tested our model and the topology-discovery protocol in a real environment. Figure 8 shows 17 Sun SPOT sensor nodes (circles) and a base station (square) deployed in our offices. Each node counts and stores locally the number of incoming and outgoing messages as well as the sizes of the messages in bytes.

We have repeated the following experiment 10 times:

1. Disseminate a query using simple flooding ($p = 1$) to determine the number of nodes that can be reached. This is necessary, because in real deployments external factors such as metal doors or electrical devices can prevent nodes from being reached in any case.

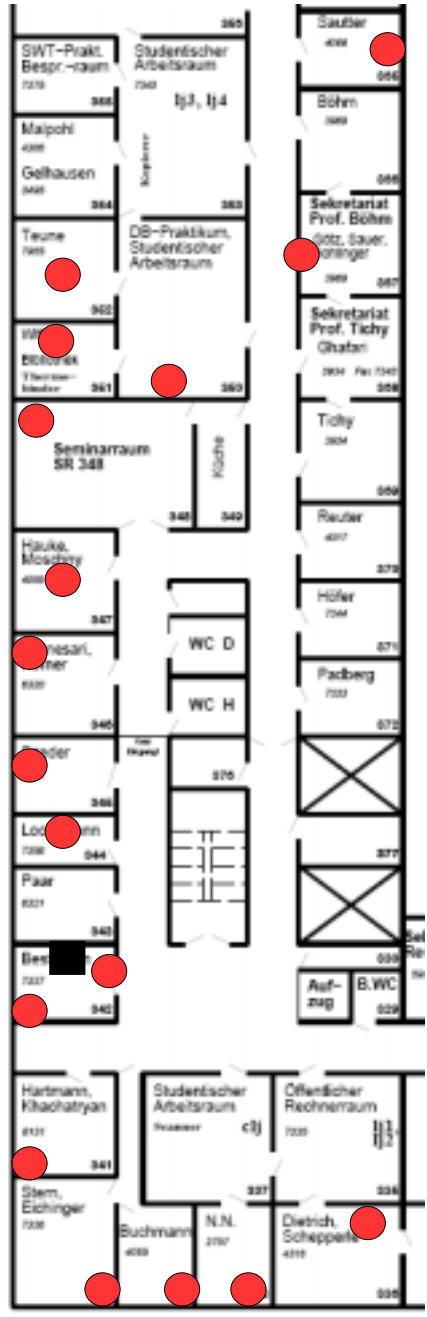


FIGURE 8
Map of 17 Sun SPOTs and a Base Station deployed at the IPD.

Flooding	Avg. Reached Nodes (of 17)	Messages Sent	Messages Received
Simple	16.3	16.3	63.8
Probabilistic	15.4	10.2	34

TABLE 2
Result of the flooding experiment using the Sun SPOT deployment

2. Fetch the topology data using the topology-discovery protocol.
3. Predict the number of nodes reached with different values for the rebroadcast probability p based on the topology information collected. Determine p_0, the lowest rebroadcast probability where a reachability of 100% is predicted.
4. Disseminate a query message into the network using probabilistic flooding with a rebroadcast probability of p_0.

Table 2 shows the average results for the 10 experiments: Generally, the accuracy of the prediction and thereby the number of nodes reached with p_0 is good, even though there is a small difference between the 16.3 nodes reached by simple flooding compared to the probabilistic flooding with 15.4 nodes reached on average. The nodes that were not reached were always the two nodes in the north east corner of the building, which have only one link to the rest of the network. With simple flooding, external factors like metal doors or electric devices might prevent these nodes from obtaining the query. Probabilistic flooding further decreases the probability that these nodes receive the query by $(1 - p)$.

Furthermore, Table 2 confirms that the number of messages sent and received with probabilistic flooding is much lower than the one with the simple flooding. The amount of energy saved due to reduced communication clearly outweighs the small inaccuracy of the prediction. Since wireless sensor networks cannot guarantee 100% reachability anyway, a small deviation in the prediction of the number of nodes reached does not limit the applicability of our approach.

5 CONCLUSION

Realizing energy-efficient query dissemination in sensor networks is challenging, even more if a predictable reachability and energy usage is required. In general, unnecessary transmissions should be avoided to save energy. It requires knowledge of the sensor network to find out which transmissions are actually required. However, obtaining this information comes with a communication overhead.

In this paper we have used probabilistic flooding as a model to explore the relations between (1) energy consumption of the query-dissemination phase, (2) the number of nodes reached and (3) the energy spent to gather structural information about the network which is required to parametrize probabilistic flooding. In particular, we have introduced an analytical model that enables the base station to estimate the reachability and energy consumption of probabilistic flooding based on connectivity information. Furthermore, we have shown how to gather this information efficiently, and we have computed the break-even point between energy saved and energy spent to obtain structural information. Both experiments with a simulator and an implementation with a testbed consisting of 17 SUN Spot nodes validate our findings.

REFERENCES

[1] Xbow technology inc. wireless sensor networks. URL http://www.xbow.com.

[2] SUN Microsystems Inc., Small Programmable Object Technology (SPOT). http://www.sunspotworld.com.

[3] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 2002.

[4] Y. Yao and J. Gehrke. Query processing in sensor networks. 2003.

[5] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 2003.

[6] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 2005.

[7] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record*, 2002.

[8] Katia Obraczka, Kumar Viswanath, and Gene Tsudik. Flooding for reliable multicast in multi-hop ad hoc networks. 2001.

[9] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999.

[10] David Simplot-Ryl, Ivan Stojmenovic, and Jie Wu. *Energy efficient backbone construction, broadcasting, and area coverage in sensor networks*. 2005.

[11] Hyojun Lim and Chongkwon Kim. Multicast tree construction and flooding in wireless ad hoc networks. In *MSWIM '00: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, 2000.

[12] Wei Peng and Xi-Cheng Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, 2000.

[13] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. 1990.

[14] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2002.

[15] B. Krishnamachari, S. Wicker, R. Bejar, M. Pearlman, and C. Critical density thresholds in distributed wireless networks. In *Communications, information and network security*, 2002.

[16] Deepak Ganesan, Bhaskar Krishnamachari, Alec Woo, David Culler, Deborah Estrin, and Stephen Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical report, 2002.

[17] R.R. Choudhury, S. Bandyopadhyay, and K. Paul. A distributed mechanism for topology discovery in ad hoc wireless networks using mobile agents. *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, 2000.

[18] Saurabh Mehta, Won-Sik Yoon, Seung-Wook Min, and Shaokai Yu. Topology generation algorithms for home sensor networks. *Software Technologies for Future Embedded and Ubiquitous Systems, 2004. Proceedings. Second IEEE Workshop on*, May 2004.

[19] A. Ahmed and B.H. Far. Topology discovery for network fault management using mobile agents in ad-hoc networks. *Electrical and Computer Engineering, 2005. Canadian Conference on*, May 2005.

[20] Ernest J. H. Chang. Echo algorithms: Depth parallel operations on general graphs. *IEEE Transactions on Software Engineering*, 8(4): 391–401, July 1982.

[21] Markus Bestehorn. The Karlsruhe Sensor Networking Simulator (KSN). http://www.ipd.uni-karlsruhe.de/KSN.

[22] Simon Kellner, Mario Pink, Detlev Meier, and Erik-Oliver Blaß. Towards a realistic energy model for wireless sensor networks. In *5th Conference on Wireless On Demand Network Systems and Services*, January 2008.

[23] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Proc. 9th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2000.